LAB 2: HIDDEN MARKOV MODELS

Namita Sharma

9/22/2020

1. Build a Hidden Markov Model (HMM)

```
library("HMM")
library("entropy")
library("grid")
library("gridExtra")
library("ggplot2")

# Q1-----------------------------------------------------------
# Build a HMM
states    <- c("Z1", "Z2", "Z3", "Z4", "Z5", "Z6", "Z7", "Z8", "Z9", "Z10")
emission  <- c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9", "X10")
nstates   <- length(states)

# Transition probabilities
transProb <- matrix(data=0, nrow=nstates, ncol=nstates,
                    dimnames=list(states, states))
for (i in 1:(nstates-1)) {
  # Assign equal probability to current and next state
  transProb[i, c(i, i+1)] <- 0.5
}
transProb[nstates, c(nstates, 1)] <- 0.5

# Emission probabilities
emissionprob <- matrix(data=0, nrow=nstates, ncol=nstates,
                       dimnames=list(states, emission))
for (i in 1:nstates) {
  # Chain the emissions starting with the emission Xi
  emission_chain <- c(emission[i:nstates], emission[-(i:nstates)])

  # Assign equal probability to the current, previous two and next two emissions in the chain
  emissionprob[i, emission_chain[c(9:10, 0:3)]] <- 0.2
}

# Initialize a general discrete time HMM
hmmInit <- initHMM(States=states,
                   Symbols=emission,
                   startProbs=c(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1),
                   transProbs=transProb,
                   emissionProbs=emissionprob)

knitr::kable(data.frame(States=hmmInit$States, Obs=hmmInit$Symbols),
```

```
            caption="Hidden states and emissions in the HMM")
```

Table 1: Hidden states and emissions in the HMM

States	Obs
Z1	X1
Z2	X2
Z3	X3
Z4	X4
Z5	X5
Z6	X6
Z7	X7
Z8	X8
Z9	X9
Z10	X10

```
knitr::kable(hmmInit$transProbs, caption="Transisiton probabilities")
```

Table 2: Transisiton probabilities

	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10
Z1	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Z2	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Z3	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0
Z4	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0
Z5	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0
Z6	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0
Z7	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0
Z8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0
Z9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
Z10	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

```
knitr::kable(hmmInit$emissionProbs, caption="Emission probabilities")
```

Table 3: Emission probabilities

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
Z1	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2
Z2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2
Z3	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0
Z4	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0
Z5	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0
Z6	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0
Z7	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0
Z8	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2
Z9	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2
Z10	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2

2. Simulate the HMM for 100 time steps

```
# Q2-----------------------------------------------------------
# Simulate the HMM for 100 time steps
T <- 100
simHmm <- simHMM(hmm=hmmInit, length=T)

knitr::kable(data.frame(States=simHmm$states, Obs=simHmm$observation)[1:5, ],
             caption="Actual states and observations simulated from the process")
```

Table 4: Actual states and observations simulated from the process

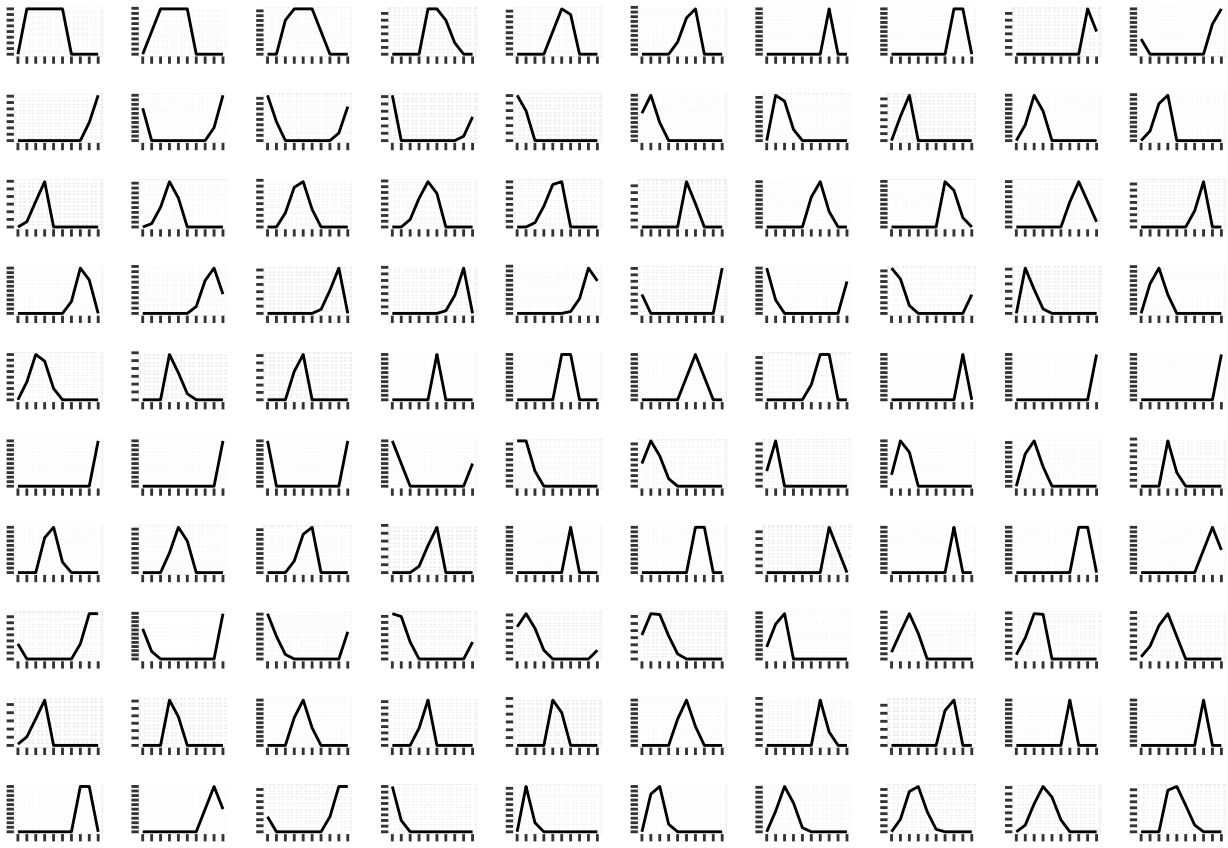
States	Obs
Z3	X4
Z3	X4
Z4	X5
Z5	X7
Z6	X5

3. Use the observations to compute the filtered and smoothed probability distributions and the most probable path

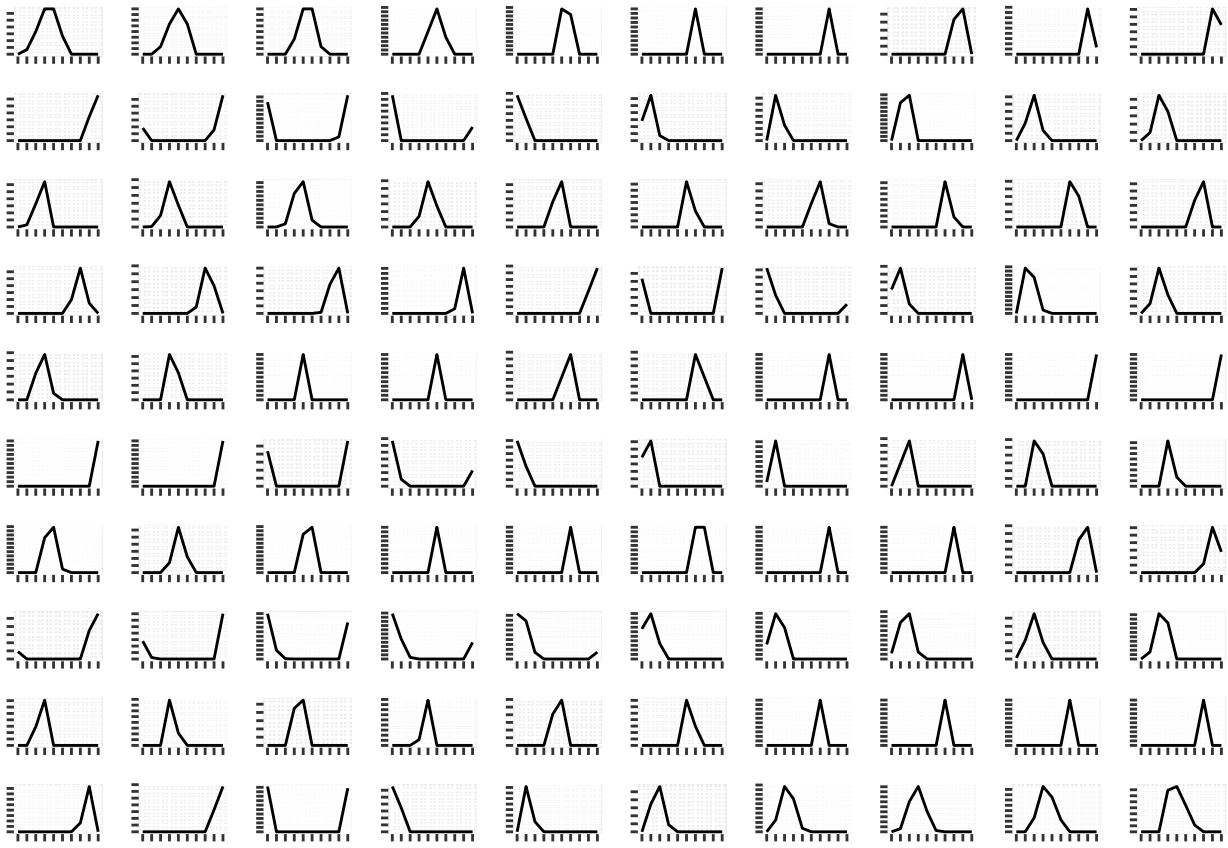
```
# Q3-----------------------------------------------------------
# Compute forward and backward probabilities
alpha  <- exp(forward(hmmInit, simHmm$observation))
beta   <- exp(backward(hmmInit, simHmm$observation))

# Compute the filtered and smoothed distributions
filtered  <- matrix(data=0, nrow=nstates, ncol=T)
smoothed  <- matrix(data=0, nrow=nstates, ncol=T)
for (i in 1:T) {
  filtered[, i] <- alpha[, i]/sum(alpha[, i])
  smoothed[, i] <- alpha[, i]*beta[, i] / sum(alpha[, i]*beta[, i])
}

# Plot the filtered and smoothed distributions for each of the data points
filter_plot <-
 lapply(X = seq_along(1:T), FUN = function(i) {
   ggplot(as.data.frame(filtered[, i]), aes(1:nstates, filtered[, i])) +
     geom_line() +
     theme(axis.title.x=element_blank(),
           axis.title.y=element_blank(),
           axis.text.x=element_blank(),
           axis.text.y=element_blank(),
           axis.ticks.x=element_line(),
           axis.ticks.y=element_line()) +
     scale_x_continuous(n.breaks=10) +
     scale_y_continuous(n.breaks=10)
 })
do.call(grid.arrange, filter_plot)
```



```
smooth_plot <-
 lapply(X = seq_along(1:T), FUN = function(i) {
   ggplot(as.data.frame(smoothed[, i]), aes(1:nstates, smoothed[, i])) +
     geom_line() +
     theme(axis.title.x=element_blank(),
           axis.title.y=element_blank(),
           axis.text.x=element_blank(),
           axis.text.y=element_blank(),
           axis.ticks.x=element_line(),
           axis.ticks.y=element_line()) +
     scale_x_continuous(n.breaks=10) +
     scale_y_continuous(n.breaks=10)
 })
do.call(grid.arrange, smooth_plot)
```



4. Compute the accuracy of hidden states guessed by each method

```
# Q4----------------------------------------------------------
# Forward-Backward algorithm
FB = function(hmmInit, observations) {

  # Compute forward and backward probabilities
  alpha  <- exp(forward(hmmInit, observations))
  beta   <- exp(backward(hmmInit, observations))

  # Filtered and Smoothed Distributions
  filtered <- prop.table(x=alpha, margin=2)
  smoothed <- prop.table(x=alpha*beta,  margin=2)

  return(list(filtered=filtered, smoothed=smoothed))
}

# Compare accuracy of different methods
compare_accuracy = function(hmmInit, observations, true_states) {

  # Forward-backward algorithm to compute the alpha and beta and the
  # filtering and smoothing distributions
  fb <- FB(hmmInit, observations)

  # Most Probable States
```

```r
  filter_MPS <- hmmInit$States[apply(X=fb$filtered, MARGIN=2, FUN=which.max)]
  smooth_MPS <- hmmInit$States[apply(X=fb$smoothed, MARGIN=2, FUN=which.max)]

  # Most Probable Path
  MPP <- viterbi(hmmInit, observations)

  # Confusion matrices
  filter_t <- table(true_states, filter_MPS)
  smooth_t <- table(true_states, smooth_MPS)
  MPP_t    <- table(true_states, MPP)

  # Accuracy
  filter_acc <- sum(diag(filter_t)) / sum(filter_t)
  smooth_acc <- sum(diag(smooth_t)) / sum(smooth_t)
  MPP_acc    <- sum(diag(MPP_t)) / sum(MPP_t)

  results <- data.frame(Accuracy=c(filter_acc, smooth_acc, MPP_acc),
                        row.names=c("Filtered", "Smoothed", "MPP"))
  return(results)
}

# Compare the accuracy of most probable states obtained from filtering and smoothing
# distributions with the most probable path
results <- compare_accuracy(hmmInit, simHmm$observation, simHmm$states)

knitr::kable(results, caption="Accuracy of different methods")
```

Table 5: Accuracy of different methods

	Accuracy
Filtered	0.67
Smoothed	0.68
MPP	0.46

The accuracy of the smoothed distribution is the highest among the three methods.

The filtered distribution is computed using the forward probabilities i.e. the hidden states z^t at time t are conditioned only on the data observed upto time t, whereas the smoothed distribution uses both the forward and the backward probabilities i.e. the hidden states z^t at time t are conditioned on the entire sequence of observations which makes the guesses more accurate. Although being best in the sense of expected number of misclassifications, the overall alignment of the most probable states obtained from filtered and smoothed distributions can have very low or even zero likelihood.

On the other hand, viterbi algorithm computes the sequence of hidden states that maximizes the joint probability of observations given the HMM. Even though the most probable path of states has the biggest joint posterior probability, it can pass states of low marginal posterior probability (marginal posterior of states at time t) because viterbi algorithm does not minimize the expected number of classification errors and hence does not have as high accuracy as the smoothed and filtered distributions.

5. Compute accuracies for different samples

```r
# Q5----------------------------------------------------------
# Simulate the HMM for 150, 200, 250, 300, 350 time steps and compare accuracy
```

```r
# for different methods
results <- data.frame(matrix(ncol=5, nrow=3),
                      row.names=c("Filtered", "Smoothed", "MPP"))
names(results) <- c("T=150", "T=200", "T=250", "T=300", "T=350")

i <- 1
for (T in c(150, 200, 250, 300, 350)) {
  sample  <- simHMM(hmm=hmmInit, length=T)
  results[, i] <- compare_accuracy(hmmInit, sample$observation, sample$states)
  i <- i+1
}

knitr::kable(results, caption="Accuracy of different methods")
```

Table 6: Accuracy of different methods

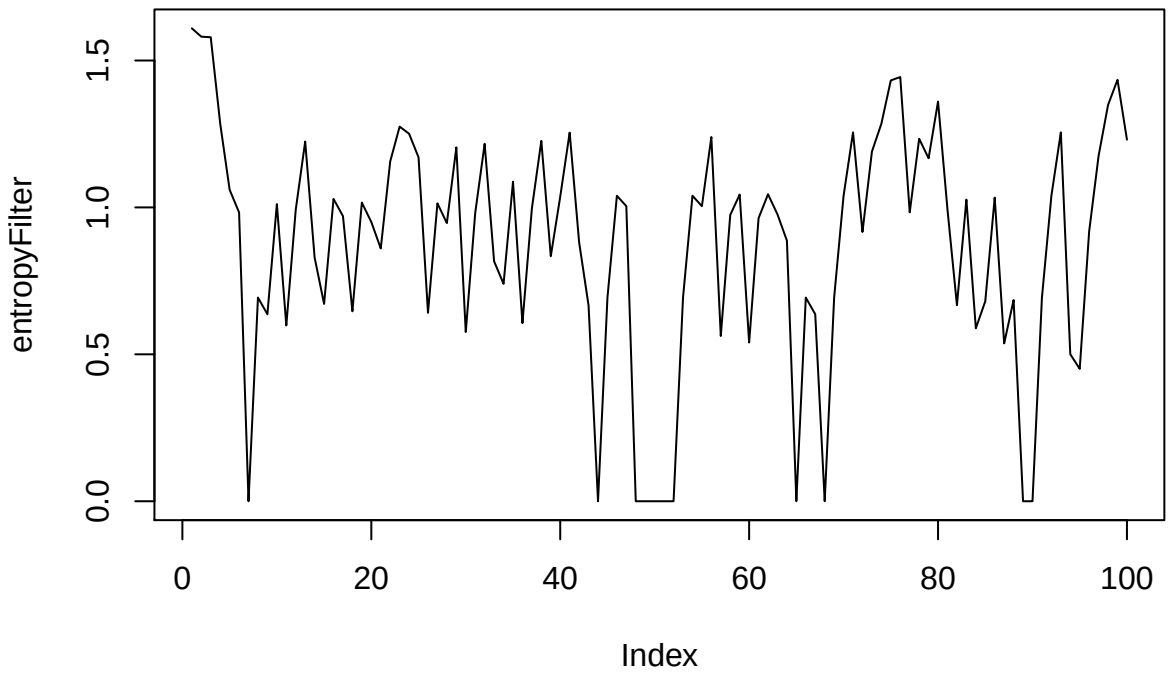
	T=150	T=200	T=250	T=300	T=350
Filtered	0.5400000	0.485	0.540	0.5266667	0.5028571
Smoothed	0.7333333	0.615	0.704	0.7200000	0.6800000
MPP	0.5000000	0.490	0.480	0.5666667	0.6142857

6. Do more observations help to know better where the robot is?

Entropy is a measure of the uncertainty in tracking the actual process (i.e.the robot) given a hidden markov model and a sequence of observations. It is seen that the entropy of the filtered distribution does not decrease with increase in time t and number of observations. Hence it cannot be said that the tracking gets better with more observations.

It was noticed however that the overall entropy decreased from 1.5 to 1.2 when using the smoothed distribution to get the trajectory of the robot. But the 'online' tracking is seen not to improve for the given markov model even with more observations. It may be possible to improve it by learning the true parameters of the HMM through EM or other optimization algorithms.

```r
# Q6-------------------------------------------------------
# Entropy
fb <- FB(hmmInit, simHmm$observation)
entropyFilter <- apply(X=fb$filtered, MARGIN=2, FUN=entropy.empirical)
graphics::plot(entropyFilter, type="l")
```



7. Compute the probabilities for hidden states for time step 101

The probabilities of the hidden states for time step t=101 can be computed using the posterior probabilities of hidden states at time t=100 given all the observations (i.e. the smoothed distribution at time t=100) and multiplying it with the underlying transition probabilities between the states.

NOTE : We get the same posterior of the hidden states at time t=100 with both smoothed and filtered distributions since we condition on the same number of observations (i.e. the entire sequence of observed data)

```
# Q7-------------------------------------------------------------
# Prediction
postState <- posterior(hmmInit, simHmm$observation)
z_101 <- postState[, 100] %*% transProb

knitr::kable(z_101, caption="Probablities of hidden states for t=101")
```

Table 7: Probablities of hidden states for t=101

Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10
0	0	0	0.172956	0.3616352	0.2987421	0.1383648	0.0283019	0	0